natives for use in documents that do not use, accept, or process ink or speech data and that do not support a data structure including such alternatives. More specifically, aspects of the present invention can be used to enable corrections in any Edit or RichEdit field, any sub-classed Edit or RichEdit field, or any super-classed Edit or RichEdit field, without replacing any existing Edit or RichEdit binary file. Because Edit and RichEdit fields are commonly used throughout the Windows® operating system (available from Microsoft Corporation) and existing application programs, the ability to support editing and corrections in these fields provides very good coverage of existing programs and fields (for example, perhaps 90% of fields). Additionally, at least some examples of the present invention allow users to turn off the functionality, if desired, e.g., at the application program level, edit class level, or specified field level, so the user need not use aspects of the invention or make it available, if desired (e.g., in the event that aspects of the invention are believed to interfere in some manner with proper operation of a specific application program and/or in specific documents).

[**0087**] Basically, systems and methods according to this specific example and implementation of the invention operate as follows. Insertion point location changes and selection events are monitored (as described in more detail below) using the Microsoft Active Accessibility framework that is part of Microsoft's Windows XP Tablet PC Edition™ operating system. Detection of these changes and events allow determination of a new insertion point location and/or a selection event. These systems and methods use conventional Windows® Edit messages (supported by Edit and RichEdit Windows® classes) to determine the insertion point location or text selection within the electronic document and the document's length. This information is used to infer the type of text altering event that occurred in almost all common operations. By tracking the insertion point location changes and selection events, systems and methods according to at least this example of the invention are able to maintain a separate document (like the expanded version of electronic document **714**) that mirrors the content of the user-visible document (e.g., a document that does not support the Text Services Framework, like electronic document **706**). The separate expanded version of the electronic document is a full Text Services Framework document, so it supports and maintains the alternatives for recognized words injected into the document, e.g., by speech or handwriting recognition or another source. This full Text Services Framework document then may be used by systems and methods according to this example of the invention to provide alternatives for a selected word to present to the user in an appropriate correction user interface, as generally described above.

[**0088**] Various aspects of the above noted systems and methods will be described in more detail. The Text Services Framework on commercially available pen-based computing systems using Microsoft's Windows XP Tablet PC Edition™ operating system contain a mode of operation that supports text injection into existing, non-Text Services Framework type applications. This "text injector" system is known as "Text Services Framework Unaware Application Support," and it is enabled for various 32 bit applications. A Text Services Framework text insertion product will receive notification that a Text Services Framework supported document is active when focus switches to a Text Services Framework Unaware Application Support application. When the text insertion product wants to inject text (e.g., into the Text Services Framework Unaware Application Support application), it injects the text into a Text Services Framework supported document (e.g., the Text Services Framework Unaware Application Support document) in the same way it would if the document was a full Text Services Framework supported document. The Text Services Framework Unaware Application Support then takes that text and causes it to be injected into the actual field of the non-Text Services Framework document.

[**0089**] Because the Text Services Framework Unaware Application Support document is, for all intents and purposes, a full but temporary context, all the usual behaviors of a full Text Services framework document are available when it is used. Therefore, any Text Services Framework text insertion product can monitor this context and receive notifications when it changes. In systems and methods according to at least some examples of this invention, when a text injection into the Text Services Framework Unaware Application Support context is noted, the injection is queried and the text and all associated alternatives data (if any) are copied out (e.g., in a serialized data byte stream to the backing store document).

[**0090**] An insertion point tracking piece is used to monitor changes in the current insertion point location and/or selection events using Microsoft Active Accessibility events, to determine when to investigate changes in an electronic document. When a Microsoft Active Accessibility caret event occurs, Edit messages are used to determine the current insertion point location in the document and the document length. From this information, the type of text change that just occurred in the document often can be inferred. In the cases where the changes cannot be inferred, the expanded version of the electronic document (or the backing store version) is discarded. The following table shows examples of how various insertion point ("IP") location change events can be interpreted in various examples of systems and methods according to the invention:

Original Element	End Element	Document Length Change	IP Location at Start	IP Location at End	Comments
IP	IP	+x	P	P + x	Text added at IP Start
IP	IP	+x	P	Not P + x	Lost Synchronization
IP	IP	None	P	P	IP Location Change or Nothing Happened
IP	IP	−1	P	P − 1	Character before IP deleted by backspace
IP	IP	−1	P	P − 1 (on previous line)	Carriage return before IP deleted by backspace
IP	IP	−2	P	P − 2 (on previous line)	Carriage return/line feed pair deleted by backspace
IP	IP	−1	P	P	Following character deleted
IP	IP	−1 (−1 Line)	P	P	Following carriage return deleted
IP	IP	−2 (−1 Line)	P	P	Following carriage return/line feed pair deleted
IP	IP	All Others			Lost Synchronization